

BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	

```
PPPPPPPP      AAAAAA      BBBB8888      TTTTTTTTTT      DDDDDDDDD      RRRRRRRR      IIIIII      VV      VV      RRRRRRRR
PPPPPPPP      AAAAAA      98888888      TTTTTTTTTT      DDDDDDDDD      RRRRRRRR      IIIIII      VV      VV      RRRRRRRR
PP      PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PPPPPPPP      AA      AA      88888888      TT      DD      DD      RRRRRRRR      II      VV      VV      RRRRRRRR
PPPPPPPP      AA      AA      88888888      TT      DD      DD      RRRRRRRR      II      VV      VV      RRRRRRRR
PP      AAAAAAAAAA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      AAAAAAAAAA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      AA      AA      BB      BB      TT      DD      DD      RR      RR      II      VV      VV      RR      RR
PP      AA      AA      88888888      TT      DDDDDDDDD      RR      RR      IIIIII      VV      VV      RR      RR
PP      AA      AA      88888888      TT      DDDDDDDDD      RR      RR      IIIIII      VV      VV      RR      RR

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
```

(1)	81	DECLARATIONS
(1)	204	ACTION ROUTINES
(1)	284	STATE/ACTION TABLES
(1)	365	CI port bootstrap device initialization
(1)	867	CI port bootstrap driver Q10
(1)	989	CI port bootstrap device disconnect

```
0000 1 .TITLE PABTDRIVR - CI PORT BOOT DRIVER
0000 2 .IDENT 'V04-001'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: BOOTS
0000 31
0000 32 ABSTRACT:
0000 33 This module contains the bootstrap device driver for the
0000 34 CI port and disks accessed over it.
0000 35
0000 36 ENVIRONMENT: IPL 31, kernel mode, code must be PIC
0000 37
0000 38 AUTHOR: Kerbey T. Altmann, CREATION DATE: 20-Nov-1981
0000 39
0000 40 MODIFIED BY:
0000 41
0000 42 V04-001 WMC0001 Wayne Cardoza 05-Sep-1984
0000 43 Built in page table must allow for bad pages.
0000 44
0000 45 V03-013 KTA3200 Kerbey T. Altmann 26-Jun-1984
0000 46 Fix some bugs found during new processor testing.
0000 47
0000 48 V03-012 KTA3111 Kerbey T. Altmann 12-Mar-1984
0000 49 Add support for booting off dual-pathed disks.
0000 50 MAINT-INIT the port after doing dump.
0000 51
0000 52 V03-011 KTA3082 Kerbey T. Altmann 03-Oct-1983
0000 53 Redo startup for better VC OPEN conditions.
0000 54
0000 55 V03-010 ROW0230 Ralph O. Weber 28-SEP-1983
0000 56 Change a B^PQB_PTR offset to a W^PQB_PTR offset near label
0000 57 350$ to correct a link truncation error.
```

```
0000 58 :
0000 59 :
0000 60 : V03-009 KTA3075 Kerbey T. Altmann 29-Aug-1983
0000 61 : Make status check look at major code only - this
0000 62 : will allow booting off disk already ONLINE.
0000 63 :
0000 64 : V03-008 KDM0059 Kathleen D. Morse 13-Jul-1983
0000 65 : Replace use of IPR TODR with new TIMEDWAIT macro.
0000 66 :
0000 67 : V03-006 KTA3067 Kerbey T. Altmann 02-Jul-1983
0000 68 : Enhance UNIT_DISC.
0000 69 :
0000 70 : V03-005 KTA3057 Kerbey T. Altmann 17-Jun-1983
0000 71 : Redo for new SCS definitions. Add SET CNTRL CHAR
0000 72 : command, support for boot device name, UNIT_DISC.
0000 73 :
0000 74 : V03-004 KTA3034 Kerbey T. Altmann 02-Feb-1983
0000 75 : Recover the boot node name.
0000 76 :
0000 77 : V03-003 KTA3011 Kerbey T. Altmann 23-Sep-1982
0000 78 : Fix misc bugs, add CREDIT_RSP, loop forever if no path.
0000 79 :
0000 80 :--
0000 81 : .SBTTL DECLARATIONS
0000 82 :
0000 83 : INCLUDE FILES:
0000 84 :
0000 85 :
0000 86 : $BQODEF : Boot qio offsets
0000 87 : $BTDDDEF : Boot device types
0000 88 : $CIBDDEF : CI BDT entry def
0000 89 : $CIBHANDEF : CI Buffer handle
0000 90 : $IODEF : I/O function codes
0000 91 : $MSCPDEF : MSCP definitions
0000 92 : $NDTDEF : Adapter codes
0000 93 : $PAREGDEF : CI port registers
0000 94 : $PPDDEF : PPD layer definitions
0000 95 : $PRDEF : Processor registers
0000 96 : $PTEDEF : Page table entries
0000 97 : $RPBDEF : RPB offsets
0000 98 : $SCSDEF : SCS layer definitions
0000 99 : $SSDEF : Status codes
0000 100 : $VADEF : Virtual addresses
0000 101 : $VMBARGDEF : VMB argument list offsets
0000 102 :
0000 103 :
0000 104 : EQUATED SYMBOLS:
0000 105 :
0000 106 :
00061A80 0000 107 TIME = 400*1000 : Number of 10 micro-sec intervals
0000 108 : in a 4 second time-wait loop
00000060 0000 109 DG_SIZ = 96
00000060 0000 110 MS_SIZ = 96
0000 111
0000 112 $DEFINI PQB
0000 113
0000 114 $DEF PQB_Q_CMDQ0 .BLKQ
```

```
0008 115 $DEF PQB_Q_CMDQ1 .BLKQ
0010 116 $DEF PQB_Q_CMDQ2 .BLKQ
0018 117 $DEF PQB_Q_CMDQ3 .BLKQ
0020 118 $DEF PQB_Q_RESPQ .BLKQ
0028 119 $DEF PQB_L_DFRQ_HDR .BLKL
002C 120 $DEF PQB_L_MFRQ_HDR .BLKL
0030 121 $DEF PQB_L_DQE_LEN .BLKL
0034 122 $DEF PQB_L_MQE_LEN .BLKL
0038 123 $DEF PQB_L_VPQB_BASE .BLKL
003C 124 $DEF PQB_L_BDT_BASE .BLKL
0040 125 $DEF PQB_L_BDT_LEN .BLKL
0044 126 $DEF PQB_L_SPT_BASE .BLKL
0048 127 $DEF PQB_L_SPT_LEN .BLKL
004C 128 $DEF PQB_L_GPT_BASE .BLKL
0050 129 $DEF PQB_L_GPT_LEN .BLKL
0054 130 $DEF TAB_L_HOLE .BLKL
0058 131 $DEF TAB_Q_DFRQ .BLKQ
0060 132 $DEF TAB_Q_MFRQ .BLKQ
0068 133 $DEF TAB_L_STATE .BLKL
006C 134 $DEF TAB_L_TIMER .BLKL
0070 135 $DEF TAB_L_RSTAID .BLKL
0074 136 $DEF TAB_L_PAGETBL .BLKL
0078 137 $DEF TAB_B_BDT .BLKB 16
0088 138 $DEF TAB_L_RCONID .BLKL
008C 139 $DEF TAB_L_LCONID .BLKL
0090 140 $DEF TAB_PKT0 .BLKB MS_SIZ
00F0 141 $DEF TAB_PKT1 .BLKB MS_SIZ
0150 142 $DEF TAB_PKT2 .BLKB MS_SIZ
0180 143 $DEF TAB_PKT3 .BLKB MS_SIZ
0210 144 $DEF TAB_PKT4 .BLKB MS_SIZ
0270 145 $DEF TAB_PKT5 .BLKB MS_SIZ
02D0 146 $DEF TAB_PKT6 .BLKB MS_SIZ
0330 147 $DEF TAB_PKT7 .BLKB MS_SIZ
0390 148 $DEF TAB_PKT8 .BLKB MS_SIZ
03F0 149 $DEF TAB_PKT9 .BLKB MS_SIZ
0450 150
00000450 0450 151 TAB_LEN = .-PQB_Q_CMDQ0
0450 152
0450 153 $DEFEND
0000 154
0000 155 ;
0000 156 ; LOCAL MACROS
0000 157 ;
0000 158 .MACRO $QRETRY OPCODE,OPER1,OPER2,ERROR,?LOOP,?OK
0000 159
0000 160 CLRL R0
0000 161 LOOP: OPCODE OPER1,OPER2
0000 162 BCC OK
0000 163 AOBLSS #63,R0,LOOP
0000 164 BRW ERROR
0000 165 OK:
0000 166 .ENDM $QRETRY
0000 167
0000 168 ;
0000 169 ; OWN STORAGE:
0000 170 ;
0000 171
```

```
0000 172 :  
0000 173 : Boot driver table entry  
0000 174 :  
0000 175  
0000 176 $BOOT_DRIVER DEVTYPE = BTDSK_HSCCI,- : Device type (MSCP/CI)  
0000 177 SIZE = PA_DRVSIZ,- : Driver size  
0000 178 ADDR = START_DRV,- : Driver starting address  
0000 179 ENTRY = PA_DRIVER,- : Driver entry point  
0000 180 UNIT_INIT = PA_INIT,- : Driver unit init entry  
0000 181 UNIT_DISC = PA_DISC,- : Driver unit disconnect entry  
0000 182 DRIVERNAME = DSRDRVNAME,- : Driver disk name  
0000 183 AUXDRVNAME = PRTDRVNAME,- : Driver port name  
0000 184 DEVNAME = DEVNAME : Boot device name  
0000 185  
0000 186 START_DRV:  
0000 187 DSKDRVNAME:  
58 45 2E 52 45 56 49 52 44 55 44 00' 0000 188 .ASCIC /DUDRIVER.EXE/ : Disk class driver filename  
45 000C  
0C 0000  
58 45 2E 52 45 56 49 52 44 41 50 00' 000D 189 PRTDRVNAME:  
45 0019 190 .ASCIC /PADRIVER.EXE/ : Port driver filename  
0C 000D  
55 44 001A 191 DEVNAME:.ASCII /DU/ : Boot device name  
001C 192 TEMPL_MSG:  
0042 001C 193 .WORD SCSSC_CON_REQL : SCSSW_LENGTH  
0004 001E 194 .WORD PPDSC_SCS_MSG : PPD$W_MTYPE  
0000 0020 195 .WORD SCSSC_CON_REQ : SCSSW_MTYPE  
0003 0022 196 .WORD 3 : Credit  
00000000 0024 197 .LONG 0  
00010001 0028 198 .LONG ^X10001 : Conid  
00000001 002C 199 .LONG 1 : Min send/Status  
20 20 20 4B 53 49 44 24 50 43 53 4D 0030 200 .ASCII /MSCP$DISK /  
20 20 20 20 003C  
5F 4C 43 5F 4B 53 49 44 24 53 4D 56 0040 201 .ASCII /VMS$DISK_CL_DRV/   
52 56 52 44 004C  
00000034 0050 202 TEMPL_MSG_LEN=-TEMPL_MSG
```

```
0050 204 .SBTTL ACTION ROUTINES
0050 205 :
0050 206 : Routines
0050 207 :
0050 208 :
0050 209 ROUT_TABLE:
0050 210 COPY_SYSID: ; Start of the routines
0A 5A 1F E1 0050 211 BBC #VASV_SYSTEM,R10,10$ ; If we are in physical mode...
14 A2 7D 0054 212 MOVQ PPD$B_SYSTEMID(R2),- ; (i.e. called from bootdriver)
24 AC 0057 213 VMB$B_SYSTEMID(AP) ; then return the remote system id
40 A2 7D 0059 214 MOVQ PPD$Q_NODENAME(R2),- ; and the node name.
34 AC 005C 215 VMB$Q_NODENAME(AP)
05 005E 216 10$: RSB ; Leave
005F 217
005F 218 .ENABLE LSB
005F 219 ALLOC_DG:
05 1C 005F 220 $QRETRY REMQHI,TAB_Q_DFRQ(R7),R2,400$ ; Yank an entry from free queue
8E D5 006E 221 BVC 500$ ; None, trouble!
04F5 31 0070 222 400$: TSTL (SP)+ ; Get rid of first level return
0072 223 BRW ERROR ; Return an error status
0075 224
05 0075 225 500$: RSB ; Success
0076 226
0076 227 DISCARD:
54 08 A2 9A 0076 228 MOVZBL PPD$W_SIZE(R2),R4 ; Get software flag
51 58 A744 7E 007A 229 MOVAQ TAB_Q_DFRQ(R7)[R4],R1 ; Get que head
007F 230 $QRETRY IN$TT,(R2),(R1),400$ ; Put it back
51 0928 C344 DE 008D 231 MOVAL PA_DFRQ(R3)[R4],R1 ; Get correct register
61 01 D0 0093 232 MOVL #PA_DFRQ_M_DFRQC,(R1) ; Tell the port its there
05 0096 233 RSB
0097 234
0097 235 SEND_ACK:
00020004 8F D0 0097 236 MOVL #<PPD$C_ACK@16+PPD$C_ACK_LEN>,-
10 A2 009D 237 PPD$W_LENGTH(R2)
3C 11 009F 238 BRB 20$ ; Send it out
00A1 239
00A1 240 SEND_START:
50 10 A2 9E 00A1 241 MOVAB PPD$W_LENGTH(R2),R0 ; Set pointer
80 3E D0 00A5 242 MOVL #<PPD$C_START@16+PPD$C_START_LEN>,(R0)+
12 11 00A8 243 BRB 10$
00AA 244
00AA 245 OPEN_VC:
34 11 00AA 246 BRB OPEN_VC_CONT ; Spacing
00AC 247
00AC 248 SPIN:
45 11 00AC 249 BRB SPIN_CONT
00AE 250
00AE 251 ASSUME <.-ROUT_TABLE> LT 128
00AE 252 SEND_STACK:
40 A2 7C 00AE 253 CLRQ PPD$Q_NODENAME(R2) ; Clear out the node name field
50 10 A2 9E 00B1 254 MOVAB PPD$W_LENGTH(R2),R0 ; Set pointer
80 0001003E 8F D0 00B5 255 MOVL #<PPD$C_STACK@16+PPD$C_STACK_LEN>,(R0)+
80 80 3E DB 00BC 256 10$: MFPR #PR$_SID,(R0)+ ; Set fake system id
80 00600060 8F D0 00BF 257 CLRL (R0)+ ; Clear hi order
80 42534D56 8F D0 00C1 258 MOVL #<MS_SIZ@16!DG_SIZ>,(R0)+
80 544F4F42 8F D0 00C8 259 MOVL #^A/VMSB/, (R0)+ ; Opsys = VMS/VMB
00CF 260 MOVL #^A/BOOT/, (R0)+ ; Version
```

```
      80 7C 00D6 261      CLRQ (R0)+      ; Boot time
      80 D4 00D8 262      CLRL (R0)+
60    3E DB 00DA 263      MFPR #PRS_SID,(R0) ; Processor id
    0365 31 00DD 264 20$: BRW SEND_DG
      00E0 265
      00E0 266 OPEN_VC_CONT:
      19 B0 00E0 267      MOVW #PPD$C_SETCKT,-
    OE A2 00E2 268      ; Set to SETCKT
      3C 00E4 269      MOVZWL #<PPD$M_CST!PPD$M_NR-
      00E5 270      !PPD$M_NS>,-
10 A2 E000 8F 00E5 271      PPD$W_MASK(R2) ; Open the virtual circuit
      8000 8F 3C 00EA 272      MOVZWL #PPD$M_CST,-
      14 A2 00EE 273      PPD$W_M_VAL(R2) ; Clear all but the circuit status
    0356 31 00F0 274      BRW SEND
      00F3 275
      00F3 276 SPIN_CONT:
      00F3 277      TIMEDWAIT TIME=#33*1000,- ; 1/3 second wait
      00F3 278      INS1=<MOVL TAB_L_RSTAID(R7),TAB_L_HOLE(R7)>,-
      00F3 279      DONELBL=5$ ; Waste time
      05 0121 280      RSB
      0122 281
      0122 282      .DISABLE LSB
```

```
0122 284 .SBTTL STATE/ACTION TABLES
0122 285 :
0122 286 : STATE TABLE
0122 287 :
00000000 0122 288 CLOSED = 0
00000001 0122 289 ST_SENT = 1
00000002 0122 290 ST_RECV = 2
00000003 0122 291 OPEN = 3
0122 292
000000C3 0122 293 TIMEOUT = 3
0122 294
0122 295 .MACRO ACTION,ROUT
0122 296 .BYTE ROUT-ROUT_TABLE
0122 297 .ENDM
0122 298
0122 299 .MACRO SET_ST,STATE,FINISH
0122 300 .IF B FINISH
0122 301 .BYTE 128!STATE
0122 302 .IFF
0122 303 .BYTE 128!64!STATE
0122 304 .ENDC
0122 305 .ENDM
0122 306
0122 307 :
0122 308 : Action table
0122 309 :
0122 310 ACTION_TABLE:
0122 311 X:
0122 312 ACT1: ACTION COPY_SYSID ; Receipt of START from other side
0123 313 ACTION OPEN_VC
0124 314 ACTION SPIN
0125 315 ACT1A: ACTION ALLOC_DG
0126 316 ACTION SEND_STACK
0127 317 SET_ST ST_RECV
0128 318
0128 319 ACT2: ACTION COPY_SYSID ; Receipt of STACK from other side
0129 320 ACTION OPEN_VC
012A 321 ACTION SPIN
012B 322 ACTION ALLOC_DG
012C 323 ACTION SEND_ACK
012D 324 SET_ST OPEN,FINISH
012E 325
012E 326 ACT3A: ACTION ALLOC_DG ; Initial or timedout action -
012F 327 ACT3: ACTION SEND_START ; send the 1st START
0130 328 SET_ST ST_SENT
0131 329
0131 330 ACT4: SET_ST OPEN,FINISH
0132 331
0132 332 ACT5: SET_ST OPEN
0133 333
0133 334 ACT6: ACTION SEND_ACK
0134 335 SET_ST OPEN
0135 336
0135 337 ACT7: ACTION DISCARD
0136 338 SET_ST OPEN
0137 339
0137 340 ACT8: ACTION DISCARD
```

```
0138 341 SET_ST CLOSED
0139 342
0139 343 ACT9: ACTION COPY_SYSID
013A 344 ACTION SEND_STACK
013B 345 SET_ST ST_RECV
013C 346
013C 347 : State table
013C 348
013C 349 STATE_TABLE:
013C 350 :
013C 351 :
15 00 00 17 013C 352 : .BYTE ACT9-X, ACT1-X, ACT1-X, ACT8-X : START received
11 06 06 15 0140 353 : .BYTE ACT8-X, ACT2-X, ACT2-X, ACT6-X : STACK received
13 0F 15 15 0144 354 : .BYTE ACT8-X, ACT8-X, ACT4-X, ACT7-X : ACK received
10 03 0C 0D 0148 355 : .BYTE ACT3-X, ACT3A-X, ACT1A-X, ACT5-X : Timeout
014C 356 :
014C 357 : Dual path data
014C 358 :
00 00 014C 359 REM_NODE:
014C 360 : .BYTE 0,0 : Dual path nodes
014E 361 REM_NODE_INDEX:
01 362 : .BYTE 1 : Starting index to select REM_NODE
00 014F 363 : .BYTE 0 : Extra
```

```
.SBTTL CI port bootstrap device initialization
0150 365
0150 366
0150 367 :++
0150 368
0150 369 :Inputs:
0150 370
0150 371 R9 --> RPB
0150 372 AP --> VMB argument list
0150 373
0150 374 :Outputs:
0150 375
0150 376 R0 - status code
0150 377
0150 378 :--
0150 379 .ENABLE LSB
0150 380
0150 381 PA_INIT:
05FC 0150 382 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R10>
0152 383
000005AA'EF 5C DO 0152 384 MOVL AP,SAVE AP ; Save the VMB arg list
F1 AF 01 90 0159 385 MOVVB #1,REM_NODE_INDEX ; Initialize node index
24 A9 90 015D 386 MOVVB RPB$C_BOOTR2(R9),- ; Pick up path1 node
EA AF 0160 387 REM_NODE
25 A9 90 0162 388 MOVVB RPB$C_BOOTR2+1(R9),- ; Pick up path2 node
E6 AF 0165 389 REM_NODE+1
05 12 0167 390 BNEQ 5$ ; Something there
24 A9 90 0169 391 MOVVB RPB$C_BOOTR2(R9),- ; Duplicate path1 node
DF AF 016C 392 REM_NODE+1
03 DD 016E 393 5$: PUSHL #3 ; Try this three times
07 10 0170 394 10$: BSBB RE_INIT ; Do the complete initialization
03 50 E8 0172 395 BLBS R0,20$ ; Return with success
F8 6E F5 0175 396 SOBGTR (SP),10$ ; Failed, try again
04 0178 397 20$: RET
0179 398
0179 399 : Initialize the tables needed
0179 400
57 086C'CF 9E 0179 401 RE_INIT:MOVAB W^TABLE,R7 ; Cover the area
57 01FF C7 9E 017E 402 MOVAB 511(R7),R7 ; Set to round up
57 01FF 8F AA 0183 403 BICW #511,R7 ; to next page
67 0450 8F 00 6E 00 2C 0188 404 MOVCS #0,(SP),#0,#TAB_LEN,(R7); Zero it all out
50 38 DB 0190 405 MFPR #PR$ MAPEN,R0 ; Get the mapping status
0193 406 ASSUME RPB$C_ADPVIR EQ RPB$C_ADPPHY+4
53 5C A940 DO 0193 407 MOVL RPB$C_ADPPHY(R9)[R0],R3 ; Get correct pointer to port registers
2A 50 E8 0198 408 BLBS R0,50$ ; If virtual, skip some set up
019B 409
019B 410 : Mapping is physical
019B 411
5A 58 57 DO 019B 412 MOVL R7,R8 ; PQB PA=VA
01 1F 9C 019E 413 ROTL #VASV_SYSTEM,#1,R10 ; Set the system VA bit
1C AC C1 01A2 414 ADDL3 VMB$Q_UCODE(AP),- ; Start the page table in pre-allocated
56 20 AC 01A5 415 VMB$Q_UCODE+4(AP),R6 ; memory just beyond the ucode
55 56 DO 01A8 416 MOVL R6,R5 ; Save the PA of page table
54 4C A9 DO 01AB 417 MOVL RPB$C_PFNcnt(R9),R4 ; Get the number of page table entries
0104 C9 C0 01AF 418 ADDL RPB$C_BADPGS(R9),R4 ; Add in the bad pages
51 54 DO 01B4 419 MOVL R4,R1 ; Save
52 09 1C 9C 01B7 420 ROTL #28,S^#<<PTESC_KW!PTESM_VALID>a-28>&^XF,R2 ; Set up fake a SPT
86 52 DO 01BB 421 40$: MOVL R2,(R6)+ ; Put it in
```

```

      F8 52 D6 01BE 422      INCL R2      ; Step to next page
      54 F5 01C0 423      SOBGTR R4,40$  ; Loop until done
      1A 11 01C3 424      BRB 60$      ; Join common
      01C5 425      ;
      01C5 426      ; Mapping is virtual
      01C5 427      ;
58 57 15 09 EF 01C5 428 50$: EXTZV #VASV VPN,#VASS VPN,R7,R8 ; Get virtual page number
58 50 B948 DE 01CA 429      MOVAL @RPBSL_SVASPT(R9)[R8],R8 ; Find page table entry
58 68 09 78 01CF 430      ASHL #9,(R8),R8 ; Turn into physical address
      5A D4 01D3 431      CLRL R10 ; Initialize
51 00B8 C9 D0 01D5 432      MOVL RPBSL_SLR(R9),R1 ; Size of page table
55 00AC C9 D0 01DA 433      MOVL RPBSL_SBR(R9),R5 ; Address
      01DF 434      ;
      01DF 435      ; Set up the PQB
      01DF 436      ;
56 58 A7 7E 01DF 437 60$: MOVAQ TAB_Q_DFRQ(R7),R6 ; Point to start
52 28 A7 DE 01E3 438      MOVAL PQB_L_DFRQ_HDR(R7),R2 ; Ditto
      62 86 7E 01E7 439      MOVAQ (R6)+,(R2) ; Datagram free que header
82 5A C8 01EA 440      BISL R10,(R2)+ ; Set sys VA bit
      01ED 441      ASSUME TAB_Q_MFRQ EQ TAB_Q_DFRQ+8
      01ED 442      ASSUME PQB_L_MFRQ_HDR EQ PQB_L_DFRQ_HDR+4
      62 86 7E 01ED 443      MOVAQ (R6)+,(R2) ; Message free que header
82 5A C8 01F0 444      BISL R10,(R2)+ ; Set sys VA bit
      01F3 445      ASSUME PQB_L_DQE_LEN EQ PQB_L_MFRQ_HDR+4
82 0060 8F 3C 01F3 446      MOVZWL #DG_SZ,(R2)+ ; Datagram size
      01F8 447      ASSUME PQB_L_MQE_LEN EQ PQB_L_DQE_LEN+4
82 0060 8F 3C 01F8 448      MOVZWL #MS_SZ,(R2)+ ; Message size
      86 86 05 D0 01FD 449      MOVL #5,(R6)+ ; Set STATE to initial
86 00061A80 8F D0 0200 450      MOVL #TIME,(R6)+ ; Set timer
      0207 451      MOVL RPBSL_BOOTR2(R9),(R6)+ ; Transfer remote port
50 FF43 CF 90 0207 452      MOVB REM_NODE_INDEX,R0 ; Pick up remote port/node index
FF3C CF 50 01 8D 020C 453      XORB3 #1,R0,REM_NODE_INDEX ; Set other path for next time
86 FF35 CF40 9A 0212 454      MOVZBL REM_NODE[R0],(R6)+ ; Transfer remote port
86 55 5A C9 0218 455      BISL3 R10,R5,(R6)+ ; Store the psuedo page table
      021C 456      ASSUME PQB_L_VPQB_BASE EQ PQB_L_MQE_LEN+4
82 57 5A C9 021C 457      BISL3 R10,R7,(R2)+ ; Set virtual address of self
      0220 458      ASSUME PQB_L_BDT_BASE EQ PQB_L_VPQB_BASE+4
82 56 5A C9 0220 459      BISL3 R10,R6,(R2)+ ; Address of BDT
66 01 10 9C 0224 460      ROTL #16,#1,(R6) ; Set the valid bit
      0228 461      ASSUME PQB_L_BDT_LEN EQ PQB_L_BDT_BASE+4
      82 D6 0228 462      INCL (R2)+ ; Num of entries in BDT
      022A 463      ASSUME PQB_L_SPT_BASE EQ PQB_L_BDT_LEN+4
      82 55 D0 022A 464      MOVL R5,(R2)+ ; Set phys addr of page table
      022D 465      ASSUME PQB_L_SPT_LEN EQ PQB_L_SPT_BASE+4
      82 51 D0 022D 466      MOVL R1,(R2)+ ; And length
      0230 467      ;
      0230 468      ;
      0230 469      ; Now go thru the complicated startup sequence.
      0230 470      ;
      0230 471      ;
      63 63 D0 0230 472      MOVL PA_CNF(R3),PA_CNF(R3) ; Clear any SBI errors
04 A3 01 D0 0233 473      MOVL #PA_PMC_M_MIN,PA_PMC(R3) ; Do maint init
54 34 A9 D0 0237 474      MOVL RPBSL_IOVEC(R9),R4 ; Pick up address of IOVECTOR
54 28 A4 D0 023B 475      MOVL BQOSL_UCODE(R4),R4 ; Get address of ucode
      52 D4 023F 476      CLRL R2 ; Set control store address
14 A3 52 D0 0241 477 70$: MOVL R2,PA_MADR(R3) ; Give CS to CI
18 A3 84 D0 0245 478      MOVL (R4)+,PA_MDATR(R3) ; Write 4 bytes of ucode
```

```
14 A3 52 00001000 8F C9 0249 479 BISL3 #^X1000,R2,PA_MADR(R3) ; Set CS addr of h.o. word
      18 A3 84 3C 0252 480 MOVZWL (R4)+,PA_MADR(R3) ; Write 2 bytes h.o.
E3 52 00000C00 8F F2 0256 481 AOBLS #PA_C_WCSSIZ,R2,70$ ; Loop until loaded
04 A3 00000040 8F C8 025E 482 BISL #PA_PMC_M_PSA,PA_PMC(R3) ; Set program start addr
14 A3 00000400 8F D0 0266 483 MOVL #PA_C_UCODEST,PA_MADR(R3) ; Set it
      0924 C3 01 D0 026E 484 MOVL #PA_PIC_M_PIC,PA_PIC(R3) ; Start port
      0273 485 ;
      0273 486 ; Wait a while for port to do its thing.
      0273 487 ;
      0273 488 TIMEDWAIT TIME=TAB_L_TIMER(R7),- ; Time to wait
      0273 489 INS1=<BITL #PA_PMC_M_MIF,PA_PMC(R3)>,- ; Check status reg
      0273 490 INS2=<BNEQ 80$>,- ; Br if all done
      0273 491 DONELBL=80$ ; Label if all done
      03 50 E8 029F 492 BLBS R0,100$ ; Br if wait time not exceeded
      02C5 31 02A2 493 90$: BRW ERROR ; *** Br on yes, ERROR ***
      02A5 494
      0900 C3 08 D1 02A5 495 100$: CMPL #PA_PS_M_PIC,PA_PS(R3) ; Check we are done
      02C5 F6 12 02AA 496 BNEQ 90$ ; *** ERROR ***
      0904 C3 58 D0 02AC 497 MOVL R8,PA_PQBBR(R3) ; Set the physical addr of PQB
      04 A3 02 C8 02B1 498 BISL #PA_PMC_M_MTD,PA_PMC(R3) ; Disable MST
      091C C3 01 D0 02B5 499 MOVL #PA_PEC_M_PEC,PA_PEC(R3) ; Enable the port
      0918 C3 01 D0 02BA 500 MOVL #PA_PSR_M_PSC,PA_PSR(R3) ; Release the status register
      02BF 501 ;
      02BF 502 ; Initialization complete. Shutdown all circuits except that of our
      02BF 503 ; remote port. Now send out a REQID to remote port.
      02BF 504 ;
      52 0090 C7 DE 02BF 505 MOVAL TAB_PKT0(R7),R2 ; Set to cover DG
      55 0F D0 02C4 506 MOVL #15,R5 ; Set for max port
      70 A7 55 91 02C7 507 110$: CMPB R5,TAB_L_RSTAD(R7) ; Our remote port?
      1A 13 02CB 508 BEQL 120$ ; Yes, skip this
      0C A2 55 B0 02CD 509 MOVW R5,PPDSB_PORT(R2) ; Set port number
      B0 02D1 510 MOVW #<PPDSM_RSP08- ;
      02D2 511 !PPDSC-SETCKT>,- ;
      02D2 512 PPD$B_OPC(R2) ; Set to get it back
      10 A2 3C 02D7 513 MOVZWL #PPDSM_DQ1,- ; Inhibit receipt of datagram
      10 A2 D0 02DB 514 PPD$W_MASK(R2) ;
      14 A2 D0 02DD 515 MOVL PPD$W_MASK(R2),- ; Copy mask to value
      0169 30 02E2 517 BSBW SENDX ; Set the circuit
      26 11 02E5 518 BRB LOOP ; and wait for response
      02E7 519
      DD 55 F4 02E7 520 SETCKT: ;
      02EA 521 120$: SOBGEQ R5,110$ ; Loop thru them all
      02EA 522 ;
      02EA 523 ; Now give 5 of the 6 datagrams to the port to put on the free queue.
      02EA 524 ; By giving it 6 we can be assured that 3 should always be on the actual
      02EA 525 ; queue and not hidden away in internal caches.
      02EA 526 ;
      55 09 D0 02EA 527 MOVL #9,R5 ; Keep one in reserve
      04 55 91 02ED 528 130$: CMPB R5,#4 ; First four are messages
      04 15 02F0 529 BLEQ 132$ ;
      08 A2 01 D0 02F2 530 MOVL #1,PPD$W_SIZE(R2) ; Show as message
      FD7D 30 02F6 531 132$: BSBW DISCARD ; Put it on free queue
      52 60 A2 9E 02F9 532 MOVAB DG_SIZ(R2),R2 ; Step to next one
      ED 55 F5 02FD 533 SOBGTR R5,130$ ;
      0300 534 ;
      0300 535 ; Now send out a REQID to the remote node to get the handshake started.
```

```
0300 536 ; This will also confirm that the other side is there
0300 537 ;
05 B0 0300 538 MOVW #PPD$C_REQID,-
OE A2 0302 539 PPD$B-OPC(R2) ; Request ID from target
10 A2 7C 0304 540 CLRQ PPD$Q_XCT_ID(R2)
013F 30 0307 541 BSBW SEND ; Send it out and wait
68 A7 D4 030A 542 CLRL TAB_L_STATE(R7) ; Set state to closed
030D 543 ;
030D 544 ;*****
030D 545 ;
030D 546 ; Wait loop and pseudo-interrupt handler
030D 547 ;
030D 548 ;*****
030D 549 ;
24 A7 D5 030D 550 LOOP: TSTL PQB_Q_RESPQ+4(R7) ; Anything already there?
59 12 0310 551 BNEQ 170$ ; Yes, take it off
0312 552 TIMEDWAIT TIME=TAB_L_TIMER(R7),- ; Time to wait
0312 553 INS1=<BITL #PA_PMC_M_MIF,PA_PMC(R3)>,- ; Check status reg
0312 554 INS2=<BNEQ 135$>,- ; Br if all done
0312 555 DONELBL=135$ ; Label if all done
0918 C3 01 D0 033E 556 MOVL #PA_PSR_M_PSC,PA_PSR(R3) ; Release status register to port
03 50 E8 0343 557 BLBS R0,T40$ ; Br if wait time not exceeded
00AA 31 0346 558 BRW TIMEOUT ; Br on yes
38 63 D1 0349 559 140$: CMPL PA_CNF(R3),#NDT$_CI ; Any config reg bits set except type?
OC 13 034C 561 BEQL 150$ ; No, okay
D0 034E 562 MOVL #<PA_CNF_M_CRD-
034F 563 !PA_CNF_M_MXTFLT-
034F 564 !PA_CNF_M_PARFLT-
034F 565 !PA_CNF_M_URDFLT-
034F 566 !PA_CNF_M_WSQFLT-
034F 567 !PA_CNF_M_XMTFLT>,-
63 EC010000 8F 034F 568 PA_CNF(R3) ; Clear 'don't care' bits
38 63 D1 0355 569 CMPL PA_CNF(R3),#NDT$_CI ; Try again
OE 12 0358 570 BNEQ 160$ ; *** ERROR ***
51 0900 C3 D0 035A 571 150$: MOVL PA_PS(R3),R1
51 FFFFFFFF 8F D3 035F 572 BITL #^C<PA_PS_M_RQA!PA_PS_M_PIC>,R1 ; Any bits but RQA or PIC?
03 13 0366 573 BEQL 170$ ; No, okay
01FF 31 0368 574 160$: BRW ERROR ; *** ERROR ***
036B 575 ;
036B 576 ; Remove the entry from the response queue.
036B 577 ;
036B 578 170$: $QRETRY REMQHI,PQB_Q_RESPQ(R7),-
036B 579 R2,ERROR ; Get next response, addr in R2
03 1C 037A 580 BVC 175$ ; Br if one
FF8E 31 037C 581 BRW LOOP ; Br if none
037F 582 ;
05 EF 037F 583 175$: EXTZV #PPD$V_STSTYP,-
03 0381 584 #PPD$S-STSTYP,-
51 OD A2 0382 585 PPD$B-STATUS(R2),R1 ; Check the status
36 13 0385 586 BEQL 180$ ; Okay
05 51 91 0387 587 CMPB R1,#PPD$C_TYPPN ; Bad, No path?
DC 12 038A 588 BNEQ 160$ ; No, something else
038C 589 TIMEDWAIT TIME=#100*1000,- ; One second wait
038C 590 INS1=<MOVL TAB_L_RSTAIID(R7),TAB_L_HOLE(R7)>,-
038C 591 DONELBL=500$ ; Waste time
FDBC 31 03BA 592 BRW RE_INIT ; Retry the whole thing
```

```
03BD 593 :  
03BD 594 : Dispatch on opcode type. The skip chain used here is very ugly but it  
03BD 595 : is shorter than any other way to pick 5 items out of a list of 40.  
03BD 596 :  
OE A2 91 03BD 597 180$: CMPB PPD$B_OPC(R2),-  
19 03C0 598 #PPD$C_SETCKT ; Is it a sent SETCKT?  
03 12 03C1 599 BNEQ 190$ ; No, skip on  
FF21 31 03C3 600 BRW SETCKT  
03C6 601  
OE A2 91 03C6 602 190$: CMPB PPD$B_OPC(R2),-  
18 03C9 603 #PPD$C_INVTC ; Is it a sent INVTC?  
03 12 03CA 604 BNEQ 200$ ; No, skip on  
022A 31 03CC 605 BRW INVTC  
03CF 606  
OC A2 91 03CF 607 200$: CMPB PPD$B_PORT(R2),-  
70 A7 03D2 608 TAB_L_RSTAID(R7) ; Is this from our friend?  
06 13 03D4 609 BEQL 220$ ; Yes accept it  
03D6 610  
FC9D 30 03D6 611 210$: BSBW DISCARD ; No, get rid of it  
FF31 31 03D9 612 BRW LOOP  
03DC 613  
OE A2 91 03DC 614 220$: CMPB PPD$B_OPC(R2),-  
02 03DF 615 #PPD$C_SNDMSG ; Is it a sent message?  
03 12 03E0 616 BNEQ 230$ ; Yes, go deal with it  
018B 31 03E2 617 BRW MSG_SNT  
03E5 618  
OE A2 91 03E5 619 230$: CMPB PPD$B_OPC(R2),-  
21 03E8 620 #PPD$C_DGREC ; Is it a received datagram?  
16 13 03E9 621 BEQL 250$ ; Yes  
E9 1F 03EB 622 BLSSU 210$ ; Not anywhere close  
OE A2 91 03ED 623 CMPB PPD$B_OPC(R2),-  
2B 03F0 624 #PPD$C_IDREC ; Is it a received ID  
05 12 03F1 625 BNEQ 240$ ; Yes, fake a time out to start up  
50 03 D0 03F3 626 TIMEOUT: MOVL #TIMEOUT,R0 ; Timed out, set code  
12 11 03F6 627 BRB 260$ ; Go figure out what to do  
03F8 628  
OE A2 91 03F8 629 240$: CMPB PPD$B_OPC(R2),-  
22 03FB 630 #PPD$C_MSGREC ; Is it a received msg?  
D8 1A 03FC 631 BGTRU 210$ ; No, ignore it  
007E 31 03FE 632 BRW SCS_MSG ; Yes, go process it  
0401 633  
50 12 A2 9A 0401 634 250$: MOVZBL PPD$W_MTYPE(R2),R0 ; Pick up type  
02 50 91 0405 635 CMPB R0,#ST_RECV ; Is it a handshake DG?  
CC 14 0408 636 BGTR 210$ ; No, ignore it  
040A 637  
040A 638 : Use the incoming DG type (or timeout) and the current state to index  
040A 639 : into the state table and determine next action to take.  
040A 640  
03 68 A7 91 040A 641 260$: CMPB TAB_L_STATE(R7),#OPEN ; Is it in a legal state?  
03 15 040E 642 BLEQ 265$ ; Yes, continue  
0157 31 0410 643 BRW ERROR ; No, leave  
0413 644  
50 68 B740 DE 0413 645 265$: MOVAL @TAB_L_STATE(R7)[R0],R0 ; Set up to index into  
50 FD1F CF40 9A 0418 646 MOVZBL STATE_TABLE[R0],R0 ; the state table to get action offset  
55 FCFF CF40 9E 041E 647 MOVAB ACTION_TABLE[R0],R5 ; Point to start in action table  
50 85 98 0424 648 270$: CVTBL (R5)+,R0 ; Pick up offset to routine  
07 19 0427 649 BLSS 280$ ; State change
```

```
FC22 CF40 16 0429 650 JSB ROUT_TABLE[R0] ; Do the routine
F4 11 042E 651 BRB 270$ ; Try for more
68 A7 50 C0 8F 8B 0430 652
28 50 06 E0 0430 653 280$: BICB3 #192,R0,TAB_L_STATE(R7) ; Set new state
FED0 31 0436 654 BBS #6,R0,VC_IS_OPEN ; We are finished
043A 655 L: BRW LOOP ; Else go wait for new arrival
043D 656
043D 657 .DISABLE LSB
043D 658
043D 659 ;*****
043D 660 ;
043D 661 ; End of wait loop and dispatcher
043D 662 ;
043D 663 ;*****
043D 664
043D 665 SEND_MSG:
OE 02 90 043D 666 MOVB #PPD$C_SNDMSG,- ; Set opcode to send message
A2 043F 667 PPD$B_OPC(R2)
0441 668
0441 669 SEND_ANY:
06 10 0441 670 BSBB SEND ; Send the CONNECT and wait
F5 11 0443 671 BRB L
0445 672
0445 673 SEND_DG:MOVW #PPD$C_SNDMSG,- ; Set opcode to send datagram
OE A2 0447 674 PPD$B_OPC(R2)
70 A7 B0 0449 675 SEND: MOVW TAB_L_RSTAD(R7),- ; Set in remote port/station id
OC A2 044C 676 PPD$B_PORT(R2)
0908 C3 01 D0 044E 677 SENDX: $QRETRY INSQTI(R2),PQB Q CMDQ0(R7),ERROR ; Tell port its there
05 045C 678 MOVL #PA_CQ0_M_CQC,PA_CQ0(R3)
0461 679 RSB
0462 680
0462 681 ;
0462 682 ; Now the virtual circuit is establised. Send out the CONNECT_REQ
0462 683 ;
0462 684 VC_IS_OPEN:
08 A2 01 D0 0462 685 MOVL #1,PPD$W_SIZE(R2) ; Set software flag
FC0D 30 0466 686 BSBW DISCARD ; Turn the DG into a free message
FBF3 30 0469 687 BSBW ALLOC DG ; Grab another datagram
7E 52 7D 046C 688 MOVQ R2,-(SP) ; Save some registers
FBA8 CF 34 28 046F 689 MOVCL #TEMPL_MSG_LEN,TEMPL_MSG,- ; Set up the CONNECT message
10 A2 0474 690 SCSSW_LENGTH-SCSSB_PPD(R2)
52 8E 7D 0476 691 MOVQ (SP)+,R2 ; Restore some registers
08 A2 01 D0 0479 692 MOVL #1,PPD$W_SIZE(R2) ; Set software flag
BE 11 047D 693 BRB SEND_MSG ; Send out a message
047F 694 ;
047F 695 ; Received a SCS message - process it depending on type.
047F 696 ;
68 A7 03 D0 047F 697 SCS_MSG:MOVL #OPEN,TAB_L_STATE(R7) ; Set us OPEN
14 A2 B1 0483 698 CMPW SCSSW_MTYPE-SCSSB_PPD(R2),-
00 0486 699 #SCSSC_CON_REQ ; Is it a CONNECT request?
12 13 0487 700 BEQL 20$ ; Yes
14 A2 B1 0489 701 CMPW SCSSW_MTYPE-SCSSB_PPD(R2),-
01 048C 702 #SCSSC_CON_RSP ; Is it a CONNECT response?
1F 12 048D 703 BNEQ 30$ ; No
03 22 A2 E8 048F 704 BLBS SCSSW_STATUS-SCSSB_PPD(R2),10$
00D4 31 0493 705 BRW ERROR ; Error out if bad status received
0496 706
```

```
FBDD 30 0496 707 10$: BSBW DISCARD ; Get rid of it to free queue
9F 11 0499 708 BRB L ; Go wait for ACCEPT request
      049B 709 ;
      049B 710 ; Received a CONNECT_REQUEST - Send back a NOLISTEN status: we do not
      049B 711 ; support any connections other than our own MSCP initiated ones. It
      049B 712 ; is probably the SCS directory poller.
      049B 713 ;
      0A B0 049B 714 20$: MOVW #SCSSC_STNOMAT,- ; No matching listener
      22 A2 049D 715 SCSSW_STATUS-SCSSB_PPD(R2)
50 18 A2 D0 049F 716 MOVL SCSSL_DST_CONID- -
      04A3 717 SCSSB_PPD(R2),R0 ; Reverse the SRC/DST connection ids
      D0 04A3 718 MOVL SCSSL_SRC_CONID- -
      04A4 719 SCSSB_PPD(R2),-
      04A4 720 SCSSL_DST_CONID- -
      04A8 721 SCSSB_PPD(R2)
      1C A2 50 D0 04A8 722 MOVL R0,SCSSL_SRC_CONID-SCSSB_PPD(R2)
      27 11 04AC 723 BRB 50$ ; Send it out
      04AE 724 ;
      14 A2 B1 04AE 725 30$: CMPW SCSSW_MTYPE-SCSSB_PPD(R2),-
      02 04B1 726 #SCSSC_ACCP_REQ ; Is it an ACCEPT request?
      27 12 04B2 727 BNEQ 60$ ; No
      04B4 728 ;
      04B4 729 ; Received a valid ACCEPT_REQUEST - turn it around to be a ACCEPT_RESPONSE.
      04B4 730 ;
      12 B0 04B4 731 MOVW #SCSSC_ACCP_RSPL,-
      10 A2 04B6 732 SCSSW_LENGTH-SCSSB_PPD(R2) ; Set new length
      B0 04B8 733 MOVW #<PPD$M_RSP$8-
      04B9 734 !PPD$C_SNDMSG>,-
      04B9 735 PPD$B_OPC(R2) ; Set to get it back
      0E A2 0102 8F B0 04BE 736 40$: MOVW #SCSSC_STNORMAL,-
      01 04C0 737 SCSSW_STATUS-SCSSB_PPD(R2) ; Set a success status
      50 18 A2 D0 04C2 738 MOVL SCSSL_DST_CONID- -
      04C6 739 SCSSB_PPD(R2),R0 ; Reverse the SRC/DST connection ids
      D0 04C6 740 MOVL SCSSL_SRC_CONID- -
      04C7 741 SCSSB_PPD(R2),-
      18 A2 1C A2 04C7 742 SCSSL_DST_CONID- -
      04CB 743 SCSSB_PPD(R2)
      1C A2 50 D0 04CB 744 MOVL R0,SCSSL_SRC_CONID-SCSSB_PPD(R2)
      18 A2 7D 04CF 745 MOVQ SCSSL_DST_CONID-SCSSB_PPD(R2),-
      0088 C7 04D2 746 TAB [RCONID(R7) ; Store them both for later use
      14 A2 B6 04D5 747 50$: INCW SCSSW_MTYPE-SCSSB_PPD(R2) ; Change to ACP_RSP
      FF62 31 04D8 748 BRW SEND_MSG ; Send it out and wait
      04DB 749 ;
      14 A2 B1 04DB 750 60$: CMPW SCSSW_MTYPE-SCSSB_PPD(R2),-
      08 04DE 751 #SCSSC_CR_REQ ; Is it an CREDIT request?
      06 12 04DF 752 BNEQ 70$ ; No
      04E1 753 ;
      04E1 754 ; Received a CREDIT_REQUEST - send out a CREDIT_RESPONSE and ignore
      04E1 755 ;
      0E B0 04E1 756 MOVW #SCSSC_CR_RSPL,-
      10 A2 04E3 757 SCSSW_LENGTH-SCSSB_PPD(R2) ; Set new length
      D7 11 04E5 758 BRB 40$ ; Send it out
      04E7 759 ;
      14 A2 B1 04E7 760 70$: CMPW SCSSW_MTYPE-SCSSB_PPD(R2),-
      0A 04EA 761 #SCSSC_APPL_MSG ; Is it an application message?
      7D 12 04EB 762 BNEQ ERROR ; No, ERROR
      04ED 763 ;
```

```
04ED 764 : Received packet was not a datagram or a SCS control message. It must
04ED 765 : therefor be a MSCP packet.
04ED 766 :
05A6'CF 52 D0 04ED 767 MOVL R2,W^NXT_MSG ; Hold for later
05A2'CF 57 D0 04F2 768 MOVL R7,W^PQB_PTR
04F7 769 ASSUME MSCPSV_ST_MASK EQ 0
50 FFEO 8F AB 04F7 770 BICW3 #^C<<15MSCP$S_ST_MASK>-1>,-
2A A2 04FB 771 MSCPSW_STATUS-SCSSB_PPD(R2),R0 ; Any drive errors?
19 13 04FE 772 BEQL 100$ ; No, continue on
04 50 B1 0500 773 CMPW R0,#MSCPSK_ST_AVLBL ; Yes, is it drive available?
05 13 0503 774 BEQL 80$ ; Yes
03 50 B1 0505 775 CMPW R0,#MSCPSK_ST_OFFLN ; No, is it no such drive?
60 12 0508 776 BNEQ ERROR ; No, error out
64 A9 B5 050A 777 80$: TSTW RPB$W_UNIT(R9) ; Is this 1st try at the shadow unit?
07 18 050D 778 BGEQ 90$ ; No, ordinay failure
28 A9 B0 050F 779 MOVW RPB$L_BOOTR3(R9),- ; Yes, replace unit with physical
64 A9 0512 780 RPB$W_UNIT(R9)
14 11 0514 781 BRB 110$ ; and try the ONLINE again
0516 782
FC60 31 0516 783 90$: BRW RE_INIT ; Failure, start from scratch
0519 784
28 A2 91 0519 785 100$: CMPB MSCPSB_OPCODE-SCSSB_PPD(R2),-
84 8F 051C 786 #MSCPSK_OP_STCON!-
051E 787 MSCPSK_OP_END ; Is it SET CTRL CHAR?
24 12 051E 788 BNEQ 120$ ; No
0520 789
0520 790 : After receipt of the end packet from the SET CTRL CHAR, it is time to
0520 791 : attempt to put the unit online. First we check to see if we are booting
0520 792 : from a shadowed disk. If so, we attempt the ONLINE to the shadowed unit.
0520 793 : If that fails, we try the physical unit. If we are booting from a single
0520 794 : physical unit, we do nothing special.
0520 795
50 2A A9 B0 0520 796 MOVW RPB$L_BOOTR3+2(R9),R0 ; Pick up possible shadow unit
04 18 0524 797 BGEQ 110$ ; Not a shadow unit
64 A9 50 B0 0526 798 MOVW R0,RPB$W_UNIT(R9) ; Use the shadow unit first
55 20 A2 DE 052A 799 110$: MOVAL -SCSSB_PPD(R2),R5 ; Set R5 to cover packet
85 01 D0 052E 800 MOVL #1,(R5)+ ; Set command ref number
85 64 A9 3C 0531 801 MOVZWL RPB$W_UNIT(R9),(R5)+ ; Put unit number in cmd packet field
85 09 9A 0535 802 MOVZBL #MSCPSK_OP_ONLIN,(R5)+ ; Set opcode to bring drive online
85 7C 0538 803 CLRQ (R5)+ ; Clear byte count, buff desc
85 7C 053A 804 CLRQ (R5)+ ; buff desc and LBN
65 7C 053C 805 CLRQ (R5) ; resvd and copy speed
32 B0 053E 806 MOVW #<-SCSSB_PPD-PPD$C_LENGTH>+MSCPSW_SHDW_UNIT+4,-
10 A2 0540 807 SCSSW_LENGTH-SCSSB_PPD(R2) ; Set the message length
51 11 0542 808 BRB 0 ; Send it out
0544 809
28 A2 91 0544 810 120$: CMPB MSCPSB_OPCODE-SCSSB_PPD(R2),-
89 8F 0547 811 #MSCPSK_OP_ONLIN!-
1B 12 0549 812 MSCPSK_OP_END ; Is it ONLINE?
0549 813 BNEQ 130$ ; No
054B 814
054B 815 : Received packet was a successful ONLINE end packet. Pick up the device
054B 816 : name from the MEDIA_ID field.
054B 817
16 EF 054B 818 EXTZV #MSCPSV_MTYD D1,-
05 05 054D 819 #MSCP$S_MTYD D1,-
51 3C A2 054E 820 MSCPSL_MEDIA_ID- -
```

```
51 51 08 78 0551 821          SCSSB PPD(R2),R1          ; Pull out 2nd device character
1B EF 0551 822          ASHL #8,R1,R1          ; Stick it in high byte
05 0555 823          EXTZV #MSCP$V_MTYPE DO,-
52 3C A2 0557 824          #MSCP$S_MTYPE DO,-
0558 825          MSCP$L_MEDIA_ID,-
055B 826          SCSSB PPD(R2),R2          ; Pull out 1st device character
51 4040 8F A8 055B 827          BISW #^X4040,R1          ; Make ASCII characters
FAB4 CF 52 51 A9 0560 828          BISW3 R1,R2,DEVNAME          ; Set into driver name
0566 829          ;
0566 830          ; Transfer is complete. Return with success status code.
0566 831          ;
50 01 3C 0566 832 130$: MOVZWL #SS$_NORMAL,R0          ; Set completion code
05 0569 833          RSB          ; And return
056A 834          ;
056A 835          ; Error occurred during transfer. Return and retry.
056A 836          ;
50 0054 8F 3C 056A 837 ERROR: MOVZWL #SS$_CTRLERR,R0          ; Set failure status
05 056F 838          RSB          ; Return to BOOTDRIVR
0570 839          ;
0570 840          ;
0570 841          ; Now set the controller characteristics
0570 842          ;
55 0E A2 B4 0570 843 MSG_SNT:CLRW PPD$B OPC(R2)          ; Clear response flags
20 A2 DE 0573 844          MOVAL -SCSSB_PPD(R2),R5          ; Set R5 to cover packet
85 01 D0 0577 845          MOVL #1,(R5)+          ; Set command ref number
85 85 D4 057A 846          CLRL (R5)+          ; Reserved field
85 04 9A 057C 847          MOVZBL #MSCP$K_OP_STCON,(R5)+          ; Set opcode to SET CTRL CHAR
85 85 D4 057F 848          CLRL (R5)+          ; Version & flags
85 FF 8F 9A 0581 849          MOVZBL #255,(R5)+          ; Set time out to 255 seconds
85 85 7C 0585 850          CLRL (R5)+          ; Clear time and date
65 D4 0587 851          CLRL (R5)          ; Clear controller depend params
04 B0 0589 852          MOVW #PPD$C_SCS MSG,-          ; Set PPD type to application
12 A2 B0 058B 853          PPD$W_MTYPE(R2)          ; message
14 A2 B0 058D 854          MOVW #SCSSC_APPL MSG,-          ; Set SCS type to application
2E B0 058F 855          SCSSW_MTYPE-SCSSB_PPD(R2) ; message
10 A2 B0 0591 856          MOVW #<-SCSSB_PPD-PPD$C_LENGTH>+MSCP$Q_TIME+12,-
16 A2 01 B0 0593 857          SCSSW_LENGTH-SCSSB_PPD(R2) ; Set the message length
0088 C7 7D 0595 858 Q: MOVW #1,SCSSW_CREDIT-SCSSB_PPD(R2); Always give back one credit
18 A2 B0 0599 859          MOVQ TAB_L_RCONID(R7),-
FE9B 31 059D 860          SCSSC_DST_CONID-SCSSB_PPD(R2) ; Set correct xct_id
05A2 861          BRW SEND_MSG          ; Send it out and wait
05A2 862          ;
00000000 05A2 863 PQB_PTR:.LONG 0
00000000 05A6 864 NXT_MSG:.LONG 0
00000000 05AA 865 SAVE_AP:.LONG 0
```

```
05AE 867 .SBTTL CI port bootstrap driver Q10
05AE 868
05AE 869 :++
05AE 870
05AE 871 : Inputs:
05AE 872
05AE 873 R3 - base address of adapter's register space
05AE 874 R5 - lbn for current piece of transfer
05AE 875 R6 - contains 0
05AE 876 R8 - size of transfer in bytes
05AE 877 R9 - address of the RPB
05AE 878 R10 - starting address of transfer (byte offset in first
05AE 879 page ORed with starting map register number)
05AE 880
05AE 881 FUNC(AP)- I/O operation (IOS$ READLBLK or IOS$ WRITELBLK only)
05AE 882 MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
05AE 883
05AE 884 : Outputs:
05AE 885
05AE 886 R0 - status code
05AE 887 SSS$_NORMAL - successful transfer
05AE 888 SSS$_CTRLERR - fatal controller error
05AE 889
05AE 890 R3 - must be preserved
05AE 891
05AE 892 : NOTE:
05AE 893 This routine can be called with four combinations of mapping:
05AE 894
05AE 895 1) MODE(AP) = physical and PR$_MAPEN = physical. This is the case
05AE 896 when being called from BOOTDRIVR. We use the made up page
05AE 897 table that maps VA = PA for both the port and BDT page tables.
05AE 898
05AE 899 2) MODE(AP) = virtual and PR$_MAPEN = physical. This is the case
05AE 900 when being called from SYSBOOT to read in SYS.EXE using
05AE 901 the real system page table. We continue to use the made up
05AE 902 page table for the port, but use the real system page table
05AE 903 for the BDT.
05AE 904
05AE 905 3) MODE(AP) = virtual and PR$_MAPEN = virtual. This is the case
05AE 906 when being called from BUGCHECK to read in the non-resident
05AE 907 portion of the bugcheck code. We use the real system page
05AE 908 table for both the port and the BDT.
05AE 909
05AE 910 4) MODE(AP) = physical and PR$_MAPEN = virtual. This is the case
05AE 911 when being called from BUGCHECK to write out all of physical
05AE 912 memory from the descriptors in the RPB. We use the real
05AE 913 system page table for the port, but make up a new pagetable
05AE 914 for the BDT. This table is only a page long since the max
05AE 915 IOSIZE is 127 pages. This table is created on the fly with
05AE 916 the first entry being the first PFN to be written out.
05AE 917 :--
05AE 918
00000010 05AE 919 FUNC = 16
00000014 05AE 920 MODE = 20
05AE 921
05AE 922 PA_DRIVER:
05AE 923 ; CI/HSC device driver.
```

```
05AE 924      .ENABLE LSB
05AE 925      :
05AE 926      : Translate the I/O function code into a device-dependent function
05AE 927      : code for this disk.
05AE 928      :
05AE 929      :
54 57 F1 AF DO 05AE 930      MOVL PQB_PTR,R7      ; Cover the PQB etc
68 A7 10 AC DO 05B2 931      MOVL FUNC(AP),TAB_L_HOLE(R7) ; Temp store function
52 E8 AF DO 05B7 932      MOVL #4,TAB_L_STATE(R7) ; Show we are in message
78 54 5A FE00 8F AB 05BB 933      MOVL NXT_MSG,R2 ; Pick up message buffer
54 5A 54 8000 8F A9 05BF 934      ASSUME MSCPSL_BYTE_CNT EQ MSCPSB_OPCODE+4
51 50 A9 80000000 8F C9 05BF 935      BICW3 #^C<VASM_BYTE>,R10,R4 ; Get the byte offset
35 14 AC E8 05C5 936      BISW3 #^X8000,R4,-
7C A7 58 DO 05CC 937      TAB_B_BDT+CIBD$W_FLAGS(R7) ; Set the boff and valid
54 5A 15 09 EF 05CC 938      MOVL R8,TAB_B_BDT+CIBD$L_BLEN(R7) ; Set the byte count
80000000 8F C9 05D0 939      EXTZV #VASS_VPN,#VASS_VPN,R10,R4 ; Pick up page table offset
35 14 AC E8 05D5 940      BISL3 #VASM_SYSTEM,RPB$L_SVASPT(R9),R1 ; Virtual
05DE 941      BLBS MODE(AP),30$ ; Pick correct page table
05E2 942      :
05E2 943      : CASE 1 or 4: MODE = physical
05E2 944      :
51 74 A7 DO 05E2 945      MOVL TAB_L_PAGETBL(R7),R1 ; Physical
50 38 DB 05E6 946      MFPR #PR$_MAPEN,R0 ; Check the current addr mode
2B 50 E9 05E9 947      BLBC R0,30$ ; Normal, skip out
05EC 948      :
05EC 949      : CASE 4: MODE = physical, PR$_MAPEN = virtual
05EC 950      : Send an INVALIDATE TRANSLATE-CACHE to port
05EC 951      :
05EC 952      : MOVW #<PPD$M_RSPa8- ; Send a INVALIDAT and
05ED 953      : !PPD$C_INVTC>,- ; set to get it back
05ED 954      : PPD$B_OPC(R2) ; but leave port alone
05F2 955      : MOVL SAVE_AP,AP ; SEND codes expects AP->VMB
05F6 956      : BRW SEND_ANY ; Do it
05F9 957      :
05F9 958      : INVTTC: ; Return
05F9 959      : CLRW PPD$B_OPC(R2) ; Clean up packet
05FC 960      : MOVAL B*PPAGTBL,R1 ; New page table
0600 961      : ASHL #-9,R8,R10 ; Number of pages
0605 962      : MOVL R1,R0 ; Copy start of table
0608 963      : BISL3 #<PTESC_KW!PTESM_VALID>,R4,(R0)+ ; Fake up a page table entry
0610 964      : INCL R4 ; Next PFN
0612 965      : SOBGTR R10,20$ ; Loop until done
0615 966      : CLRL R4 ; Use start of fake table
0617 967      :
0617 968      : All mapping done. Format the write packet and send it out
0617 969      :
0080 C7 6144 DE 0617 970      30$: MOVAL (R1)[R4],-
061D 971      : TAB_B_BDT+CIBD$L_SVAPTE(R7) ; Set it in *** TEMP ***
56 28 A2 DE 061D 972      : MOVAL MSCPSB_OPCODE-SCS$B_PPD(R2),R6 ; Point into message
86 21 9A 0621 973      : MOVZBL #MSCPSK_OP_READ,(R6)+ ; Assume read
20 54 A7 B1 0624 974      : CMPW TAB_L_HOLE(R7),#IOS_WRITEBLK ; Check for write function
04 12 0628 975      : BNEQ 40$ ; No, do read
FC A6 22 90 062A 976      : MOVW #MSCPSK_OP_WRITE,-4(R6) ; Set write function code
86 58 D0 062E 977      40$: MOVL R8,(R6)+ ; Set the byte count
86 86 D4 0631 978      : CLRL (R6)+ ; Set no offset in buffer
86 01 10 9C 0633 979      : ROTL #16,#1,(R6)+ ; Buffer name
86 0088 C7 D0 0637 980      : MOVL TAB_L_RCONID(R7),(R6)+ ; RCONID
```

PABTDRIVR
V04-001

- CI PORT BOOT DRIVER
CI port bootstrap driver Q10

G 4

15-SEP-1984 23:56:42 VAX/VMS Macro V04-00
6-SEP-1984 20:15:07 [BOOTS.SRC]PABTDRIVR.MAR;2

Page 20
(1)

66	55	D0	063C	981	MOVL	R5,(R6)	; Set the logical block number
	2E	B0	063F	982	MOVW	#<-SCS\$B_PPD-PPD\$C_LENGTH>+MSCPSL_LBN+4,-	
	10	A2	0641	983		SCS\$W_LENGTH-SCS\$B_PPD(R2)	; Set the message length
5C	FF63	CF	D0	0643	MOVL	SAVE_AP,AP	; SEND codes expects AP->VMB
	FF4A	31	0648	985	BRW	Q	; Send and wait
			064B	986			
			064B	987			

.DISABLE LSB

```
064B 989 .SBTTL CI port bootstrap device disconnect
064B 990
064B 991 :++
064B 992 : This routine disconnect the boot device after a bugcheck dump.
064B 993 : It sends an AVAIL packet to the controller, in effect doing a
064B 994 : dismount of the system device. It is designed to be called
064B 995 : only from BUGCHECK immediately after the dump has finished.
064B 996 : It assumes virtual mapping turned on.
064B 997
064B 998 : Inputs:
064B 999
064B 1000 : R9 --> RPB
064B 1001 : AP --> VMB argument list
064B 1002
064B 1003 : Outputs:
064B 1004
064B 1005 : R0 - status code
064B 1006
064B 1007 :--
064B 1008 .ENABLE LSB
064B 1009
064B 1010 PA_DISC:
008C 064B 1011 .WORD *M<R2,R3,R7>
064D 1012
57 FF51 CF D0 064D 1013 MOVL PQB_PTR,R7 ; Cover the PQB etc
52 FF50 CF D0 0652 1014 MOVL NXT_MSG,R2 ; Pick up message buffer addr
53 60 A9 D0 0657 1015 MOVL RPB$L ADPVIR(R9),R3 ; Pick up pointer to adp IO space
08 9A 065B 1016 MOVZBL #MSCP$K_OP_AVAIL,- ; Make drive AVAILABLE
28 A2 065D 1017 MSCP$B_OPCODE-SCSS$B_PPD(R2)
1A B0 065F 1018 MOVW #<-SCSS$B_PPD-PPD$C_LENGTH>+MSCP$B_OPCODE+4,-
10 A2 0661 1019 SCSS$W_LENGTH-SCSS$B_PPD(R2) ; Set the message length
FF2F 30 0663 1020 BSBW Q ; Send and wait
04 A3 01 D0 0666 1021 MOVL #PA_PMC_M_MIN,PA_PMC(R3); Do maint init to shut down port
04 066A 1022 RET
066B 1023
066B 1024 :
066B 1025 : Data area
066B 1026 :
0000066C 066B 1027 .=<.+3>8-4 ; .ALIGN LONG
0000086C 066C 1028 PPAGTBL:.BLKB 512
00000EBC 086C 1029 TABLE: .BLKB TAB_LEN+512
0EBC 1030
00000EBC 0EBC 1031 PA_DRVSIZ=-START_DRV
0EBC 1032
0EBC 1033 .END
```

PABTDRIVR
Symbol table

- CI PORT BOOT DRIVER

I 4

15-SEP-1984 23:56:42 VAX/VMS Macro V04-00
6-SEP-1984 20:15:07 [BOOT.SRC]PABTDRIVR.MAR;2

Page 22
(1)

STABLE	= 00000000	R	02
ACT1	00000122	R	03
ACT1A	00000125	R	03
ACT2	00000128	R	03
ACT3	0000012F	R	03
ACT3A	0000012E	R	03
ACT4	00000131	R	03
ACT5	00000132	R	03
ACT6	00000133	R	03
ACT7	00000135	R	03
ACT8	00000137	R	03
ACT9	00000139	R	03
ACTION_TABLE	00000122	R	03
ALLOC_DG	0000005F	R	03
BQO\$\$_TENUSEC	= 0000003E		
BQO\$\$_UBDELAY	= 00000042		
BQO\$\$_UCODE	= 00000028		
BTDSK_MSCCI	= 00000020		
CIBD\$\$_BLEN	= 00000004		
CIBD\$\$_SVAPTE	= 00000008		
CIBD\$\$_FLAGS	= 00000000		
CLOSED	= 00000000		
COPY_SYSID	00000050	R	03
DEVNAME	0000001A	R	03
DG_SIZ	= 00000060		
DISCARD	00000076	R	03
DSKDRVNAME	00000000	R	03
ERROR	0000056A	R	03
FUNC	= 00000010		
INVT	000005F9	R	03
IOS_WRTTELBLK	= 00000020		
L	0000043A	R	03
LOOP	0000030D	R	03
MODE	= 00000014		
MSCP\$\$_OPCODE	= 00000008		
MSCP\$\$_OP_AVAIL	= 00000008		
MSCP\$\$_OP_END	= 00000080		
MSCP\$\$_OP_ONLIN	= 00000009		
MSCP\$\$_OP_READ	= 00000021		
MSCP\$\$_OP_STCON	= 00000004		
MSCP\$\$_OP_WRITE	= 00000022		
MSCP\$\$_ST_AVLBL	= 00000004		
MSCP\$\$_ST_OFFLN	= 00000003		
MSCP\$\$_BYTE_CNT	= 0000000C		
MSCP\$\$_LBN	= 0000001C		
MSCP\$\$_MEDIA_ID	= 0000001C		
MSCP\$\$_TIME	= 00000014		
MSCP\$\$_MTYP_D0	= 00000005		
MSCP\$\$_MTYP_D1	= 00000005		
MSCP\$\$_ST_MASK	= 00000005		
MSCP\$\$_MTYP_D0	= 0000001B		
MSCP\$\$_MTYP_D1	= 00000016		
MSCP\$\$_ST_MASK	= 00000000		
MSCP\$\$_SHDW_UNT	= 00000020		
MSCP\$\$_STATOS	= 0000000A		
MSG_SNT	00000570	R	03
MS_SIZ	= 00000060		

NDTS_CI	= 00000038		
NXT_MSG	000005A6	R	03
OPEN	= 00000003		
OPEN_VC	000000AA	R	03
OPEN_VC_CONT	000000E0	R	03
PA_CNF	00000000		
PA_CNF_M_CRD	= 00010000		
PA_CNF_M_MXTFLT	= 08000000		
PA_CNF_M_PARFLT	= 80000000		
PA_CNF_M_URDFLT	= 20000000		
PA_CNF_M_WSQFLT	= 40000000		
PA_CNF_M_XMTFLT	= 04000000		
PA_CQ0	00000908		
PA_CQ0_M_CQC	= 00000001		
PA_CQ1	0000090C		
PA_CQ2	00000910		
PA_CQ3	00000914		
PA_C_UCODEST	= 00000400		
PA_C_WCSSIZ	= 00000C00		
PA_DFQ	00000928		
PA_DFQ_M_DFQC	= 00000001		
PA_DISC	0000064B	R	03
PA_DRIVER	000005AE	R	03
PA_DRVSIZ	= 00000EBC		
PA_INIT	00000150	R	03
PA_MADR	00000014		
PA_MDATR	00000018		
PA_MFQ	0000092C		
PA_MTC	00000930		
PA_MTEC	00000934		
PA_PDC	00000920		
PA_PEC	0000091C		
PA_PEC_M_PEC	= 00000001		
PA_PESR	0000093C		
PA_PFAR	00000938		
PA_PIC	00000924		
PA_PIC_M_PIC	= 00000001		
PA_PMC	00000004		
PA_PMC_M_MIF	= 00000008		
PA_PMC_M_MIN	= 00000001		
PA_PMC_M_MTD	= 00000002		
PA_PMC_M_PSA	= 00000040		
PA_PPR	00000940		
PA_PQBBR	00000904		
PA_PS	00000900		
PA_PSR	00000918		
PA_PSR_M_PSC	= 00000001		
PA_PS_M_PIC	= 00000008		
PA_PS_M_RQA	= 00000001		
PPAGTBL	0000066C	R	03
PPDSB_DEF_ST	0000001C		
PPDSB_FLAGS	0000000F		
PPDSB_HWVERS	00000034		
PPDSB_LBDATA	00000012		
PPDSB_LCB_0	00000012		
PPDSB_LCB_LPORT	00000010		
PPDSB_LCB_NPORT	0000000F		

PABTDRIVR
Symbol table

- CI PORT BOOT DRIVER

J 4

15-SEP-1984 23:56:42 VAX/VMS Macro V04-00
6-SEP-1984 20:15:07 [BOOTS.SRC]PABTDRIVR.MAR;2

Page 23
(1)

PPDSB_LCB_OPC 00000011
PPDSB_LCB_PORT 0000000E
PPDSB_OPC 0000000E
PPDSB_PORT 0000000C
PPDSB_PROTOCOL 0000001A
PPDSB_RSTATE 00000025
PPDSB_RST_PORT 00000024
PPDSB_STATUS 0000000D
PPDSB_SWFLAG 0000000B
PPDSB_SYSTEMID 00000014
PPDSB_TYPE 0000000A
PPDSC_ACK = 00000002
PPDSC_ACK_LEN = 00000004
PPDSC_DGREC = 00000021
PPDSC_IDREC = 0000002B
PPDSC_INVTC = 00000018
PPDSC_LB_LENGTH 00000046
PPDSC_LCB_DATA 00000013
PPDSC_LENGTH 00000012
PPDSC_MIN_DGSIZ 00000050
PPDSC_MSGREC = 00000022
PPDSC_REQID = 00000005
PPDSC_SCS_MSG = 00000004
PPDSC_SETCKT = 00000019
PPDSC_SNDDG = 00000001
PPDSC_SNDMSG = 00000002
PPDSC_STACK = 00000001
PPDSC_STACK_LEN = 0000003E
PPDSC_START = 00000000
PPDSC_START_LEN = 0000003E
PPDSC_TYPNP = 00000005
PPDSC_LB_LENGTH 00000046
PPDSC_LENGTH 00000012
PPDSL_BLINK 00000004
PPDSL_DG_DISC 00000028
PPDSL_FLINK 00000000
PPDSL_IN_VCD 00000018
PPDSL_LBCRC 00000042
PPDSL_PO_ACK 00000010
PPDSL_PO_NAK 00000014
PPDSL_PO_NRSP 00000018
PPDSL_P1_ACK 0000001C
PPDSL_P1_NAK 00000020
PPDSL_P1_NRSP 00000024
PPDSL_REC_BOFF 00000028
PPDSL_REC_NAME 00000024
PPDSL_RPORT_FCN 00000020
PPDSL_RPORT_REV 0000001C
PPDSL_RPORT_TYP 00000018
PPDSL_SND_BOFF 00000020
PPDSL_SND_NAME 0000001C
PPDSL_ST_ADDR 00000018
PPDSL_XCT_LEN 00000018
PPDSM_CST = 00008000
PPDSM_DQI = 00001000
PPDSM_NR = 00004000
PPDSM_NS = 00002000

PPDSM_RSP = 00000001
PPDSQ_CURTIME 00000048
PPDSQ_NODENAME 00000040
PPDSQ_SWINCARN 00000028
PPDSQ_XCT_ID 00000010
PPDSS_STSTYP = 00000003
PPDST_HWTYPE 00000030
PPDST_SWTYPE 00000020
PPDST_SWVERS 00000024
PPDSV_STSTYP = 00000005
PPDSW_LCB_LEN7 0000000C
PPDSW_LENGTH 00000010
PPDSW_MASK 00000010
PPDSW_MAXDG 0000001C
PPDSW_MAXMSG 0000001E
PPDSW_MTYPE 00000012
PPDSW_M_VAL 00000014
PPDSW_SIZE 00000008
PQB_L_BDT_BASE 0000003C
PQB_L_BDT_LEN 00000040
PQB_L_DFRQ_HDR 00000028
PQB_L_DQE_LEN 00000030
PQB_L_GPT_BASE 0000004C
PQB_L_GPT_LEN 00000050
PQB_L_MFRQ_HDR 0000002C
PQB_L_MQE_LEN 00000034
PQB_L_SPT_BASE 00000044
PQB_L_SPT_LEN 00000048
PQB_L_VPQB_BASE 00000038
PQB_PTR 000005A2 R 03
PQB_Q_CMDQ0 00000000
PQB_Q_CMDQ1 00000008
PQB_Q_CMDQ2 00000010
PQB_Q_CMDQ3 00000018
PQB_Q_RESPQ 00000020
PRS_MAPEN = 00000038
PRS_SID = 0000003E
PRTDRVNAME 0000000D R 03
PTESC_KW = 10000000
PTESM_VALID = 80000000
Q 00000595 R 03
REM_NODE 0000014C R R 03
REM_NODE_INDEX 0000014E R R 03
RE_INIT 00000179 R R 03
ROOT_TABLE 00000050 R 03
RPBSL_ADPPHY = 0000005C
RPBSL_ADPVIR = 00000060
RPBSL_BADPGS = 00000104
RPBSL_BOOTR2 = 00000024
RPBSL_BOOTR3 = 00000028
RPBSL_IOVEC = 00000034
RPBSL_PFNCT = 0000004C
RPBSL_SBR = 000000AC
RPBSL_SLR = 000000B8
RPBSL_SVASPT = 00000050
RPBSW_UNIT = 00000064
SAVE_AP 000005AA R 03

PABTDRIVR
Symbol table

- CI PORT BOOT DRIVER

K 4

15-SEP-1984 23:56:42 VAX/VMS Macro V04-00
6-SEP-1984 20:15:07 [BOOTS.SRC]PABTDRIVR.MAR;2

Page 24
(1)

SCSSB_PPD	=	FFFFFFE0		
SCSSC_ACCP_REQ	=	00000002		
SCSSC_ACCP_RSPL	=	00000012		
SCSSC_APPL_MSG	=	0000000A		
SCSSC_CON_REQ	=	00000000		
SCSSC_CON_REQL	=	00000042		
SCSSC_CON_RSP	=	00000001		
SCSSC_CR_REQ	=	00000008		
SCSSC_CR_RSPL	=	0000000E		
SCSSC_STNOMAT	=	0000000A		
SCSSC_STNORMAL	=	00000001		
SCSSL_DST_CONID	=	FFFFFFF8		
SCSSL_SRC_CONID	=	FFFFFFFC		
SCSSW_CREDIT	=	FFFFFFF6		
SCSSW_LENGTH	=	FFFFFFF0		
SCSSW_MTYPE	=	FFFFFFF4		
SCSSW_STATUS	=	00000002		
SCS_MSG		0000047F	R	03
SEND		00000449	R	03
SENDX		0000044E	R	03
SEND_ACK		00000097	R	03
SEND_ANY		00000441	R	03
SEND_DG		00000445	R	03
SEND_MSG		0000043D	R	03
SEND_STACK		000000AE	R	03
SEND_START		000000A1	R	03
SETCRT		000002E7	R	03
SIZ...	=	00000001		
SPIN		000000AC	R	03
SPIN_CONT		000000F3	R	03
SS\$_CTRLERR	=	00000054		
SS\$_NORMAL	=	00000001		
START_DRV		00000000	R	03
STATE_TABLE		0000013C	R	03
ST_RECV	=	00000002		
ST_SENT	=	00000001		
TABLE		0000086C	R	03
TAB_B_BDT		00000078		
TAB_LEN	=	00000450		
TAB_L_HOLE		00000054		
TAB_L_LCONID		0000008C		
TAB_L_PAGETBL		00000074		
TAB_L_RCONID		00000088		
TAB_L_RSTAD		00000070		
TAB_L_STATE		00000068		
TAB_L_TIMER		0000006C		
TAB_PKT0		00000090		
TAB_PKT1		000000F0		
TAB_PKT2		00000150		
TAB_PKT3		00000180		
TAB_PKT4		00000210		
TAB_PKT5		00000270		
TAB_PKT6		000002D0		
TAB_PKT7		00000330		
TAB_PKT8		00000390		
TAB_PKT9		000003F0		
TAB_Q_DFRQ		00000058		

TAB_Q_MFRQ		00000060		
TEMPL_MSG		0000001C	R	03
TEMPL_MSG_LEN	=	00000034		
TIME	=	00061A80		
TIMEOUT	=	00000003		
TIMOUT		000003F3	R	03
VASM_BYTE	=	000001FF		
VASM_SYSTEM	=	80000000		
VASS_VPN	=	00000015		
VASV_SYSTEM	=	0000001F		
VASV_VPN	=	00000009		
VC_IS_OPEN		00000462	R	03
VMBSB_SYSTEMID		00000024		
VMBSB_ARGBYTCNT		0000003C		
VMBSL_CI_HIPFN		00000030		
VMBSL_FLAGS		0000002C		
VMBSL_HI_PFN		00000010		
VMBSL_LO_PFN		0000000C		
VMBSQ_FICECACHE		00000004		
VMBSQ_NODENAME		00000034		
VMBSQ_PFNMAP		00000014		
VMBSQ_UCODE		0000001C		
X		00000122	R	03

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000944 (2372.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000028 (40.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000EBC (3772.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.10	00:00:00.74
Command processing	156	00:00:00.83	00:00:03.49
Pass 1	523	00:00:22.45	00:00:39.66
Symbol table sort	2	00:00:03.20	00:00:05.17
Pass 2	183	00:00:04.37	00:00:08.05
Symbol table output	36	00:00:00.27	00:00:00.29
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	940	00:00:31.25	00:00:57.43

The working set limit was 2000 pages.
122365 bytes (239 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2020 non-local and 71 local symbols.
1033 source lines were read in Pass 1, producing 17 object records in Pass 2.
32 pages of virtual memory were used to define 29 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SHRLIB]PALIB.MLB;1	2
_\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	3
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	10
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	23

2197 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:PABTDRIVR/OBJ=OBJ\$:PABTDRIVR MSRC\$:PABTDRIVR/UPDATE=(ENH\$:PABTDRIVR)+EXECML\$/LIB+LIB\$:BOOTS.MLB/LIB+SHRLIB\$:PALIB/LIB

0039 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY